# Flask Hands-on

## Flask Installation

The following command installs **virtualenv**

pip install virtualenv

On **Windows**, following can be used

venv\scripts\activate

We are now ready to install Flask in this environment.

pip install Flask

The above command can be run directly, without virtual environment for system-wide installation.

## Flask- Sending Form Data to a Template

The template dynamically renders an HTML table of **form** data.

Given below is the Python code of application −

from flask import Flask, render\_template, request

app = Flask(\_\_name\_\_)

@app.route('/')

def student():

return render\_template('student.html')

@app.route('/result',methods = ['POST', 'GET'])

def result():

if request.method == 'POST':

result = request.form

return render\_template("result.html",result = result)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug = True)

Given below is the HTML script of **student.html**.

<html>

<body>

<form action = "http://localhost:5000/result" method = "POST">

<p>Name <input type = "text" name = "Name" /></p>

<p>Physics <input type = "text" name = "Physics" /></p>

<p>Chemistry <input type = "text" name = "chemistry" /></p>

<p>Maths <input type ="text" name = "Mathematics" /></p>

<p><input type = "submit" value = "submit" /></p>

</form>

</body>

</html>

Code of template **(result.html)** is given below −

<!doctype html>

<html>

<body>

<table border = 1>

{% for key, value in result.items() %}

<tr>

<th> {{ key }} </th>

<td> {{ value }} </td>

</tr>

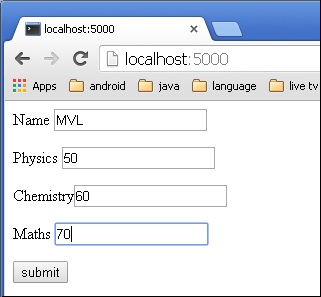
{% endfor %}

</table>

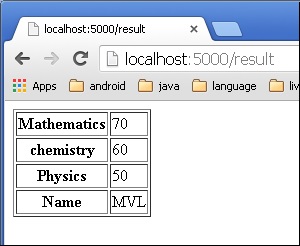
</body>

</html>

Run the Python script and enter the URL **http://localhost:5000/** in the browser.



When the **Submit** button is clicked, form data is rendered on **result.html** in the form of HTML table.



## Flask – File Uploading

The following code has **‘/upload’** URL rule that displays **‘upload.html’** from the templates folder, and **‘/upload-file’** URL rule that calls **uploader()**function handling upload process.

**‘upload.html’** has a file chooser button and a submit button.

<html>

<body>

<form action = "http://localhost:5000/uploader" method = "POST"

enctype = "multipart/form-data">

<input type = "file" name = "file" />

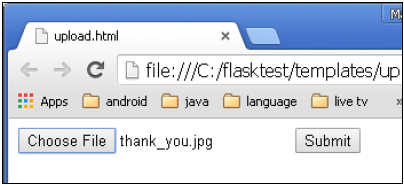
<input type = "submit"/>

</form>

</body>

</html>

You will see the screen as shown below.



Click **Submit** after choosing file. Form’s post method invokes **‘/upload\_file’**URL. The underlying function **uploader()** does the save operation.

Following is the Python code of Flask application.

from flask import Flask, render\_template, request

from werkzeug import secure\_filename

app = Flask(\_\_name\_\_)

@app.route('/upload')

def upload\_file():

return render\_template('upload.html')

@app.route('/uploader', methods = ['GET', 'POST'])

def upload\_file():

if request.method == 'POST':

f = request.files['file']

f.save(secure\_filename(f.filename))

return 'file uploaded successfully'

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug = True)

## Flask – Mail

 Run the following script in Python Shell and visit **http://localhost:5000/.**

from flask import Flask

from flask\_mail import Mail, Message

app =Flask(\_\_name\_\_)

mail=Mail(app)

app.config['MAIL\_SERVER']='smtp.gmail.com'

app.config['MAIL\_PORT'] = 465

app.config['MAIL\_USERNAME'] = 'yourId@gmail.com'

app.config['MAIL\_PASSWORD'] = '\*\*\*\*\*'

app.config['MAIL\_USE\_TLS'] = False

app.config['MAIL\_USE\_SSL'] = True

mail = Mail(app)

@app.route("/")

def index():

msg = Message('Hello', sender = 'yourId@gmail.com', recipients = ['id1@gmail.com'])

msg.body = "Hello Flask message sent from Flask-Mail"

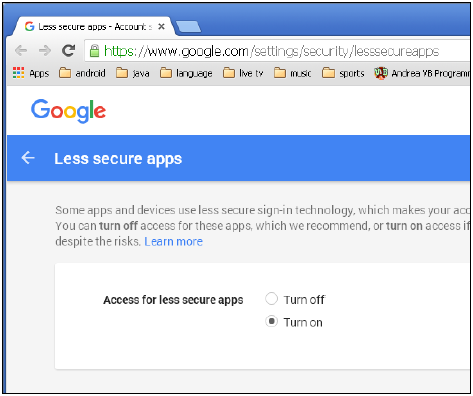
mail.send(msg)

return "Sent"

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug = True)

Note that the built-insecurity features in Gmail service may block this login attempt. You may have to decrease the security level. Please log in to your Gmail account and visit [this](https://www.google.com/settings/security/lesssecureapps) link to decrease the security.



## Flask – WTF

The design of **Contact form** is given below **(forms.py)**.

rom flask\_wtf import Form

from wtforms import TextField, IntegerField, TextAreaField, SubmitField, RadioField,

SelectField

from wtforms import validators, ValidationError

class ContactForm(Form):

name = TextField("Name Of Student",[validators.Required("Please enter

your name.")])

Gender = RadioField('Gender', choices = [('M','Male'),('F','Female')])

Address = TextAreaField("Address")

email = TextField("Email",[validators.Required("Please enter your email address."),

validators.Email("Please enter your email address.")])

Age = IntegerField("age")

language = SelectField('Languages', choices = [('cpp', 'C++'),

('py', 'Python')])

submit = SubmitField("Send")

Validators are applied to the **Name** and **Email** fields.

Given below is the Flask application script **(formexample.py)**.

from flask import Flask, render\_template, request, flash

from forms import ContactForm

app = Flask(\_\_name\_\_)

app.secret\_key = 'development key'

@app.route('/contact', methods = ['GET', 'POST'])

def contact():

form = ContactForm()

if request.method == 'POST':

if form.validate() == False:

flash('All fields are required.')

return render\_template('contact.html', form = form)

else:

return render\_template('success.html')

elif request.method == 'GET':

return render\_template('contact.html', form = form)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug = True)

The Script of the template **(contact.html)** is as follows −

<!doctype html>

<html>

<body>

<h2 style = "text-align: center;">Contact Form</h2>

{% for message in form.name.errors %}

<div>{{ message }}</div>

{% endfor %}

{% for message in form.email.errors %}

<div>{{ message }}</div>

{% endfor %}

<form action = "http://localhost:5000/contact" method = post>

<fieldset>

<legend>Contact Form</legend>

{{ form.hidden\_tag() }}

<div style = font-size:20px; font-weight:bold; margin-left:150px;>

{{ form.name.label }}<br>

{{ form.name }}

<br>

{{ form.Gender.label }} {{ form.Gender }}

{{ form.Address.label }}<br>

{{ form.Address }}

<br>

{{ form.email.label }}<br>

{{ form.email }}

<br>

{{ form.Age.label }}<br>

{{ form.Age }}

<br>

{{ form.language.label }}<br>

{{ form.language }}

<br>

{{ form.submit }}

</div>

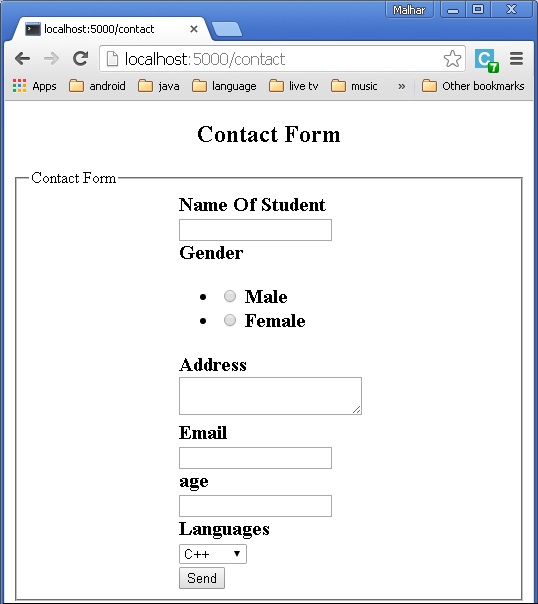
</fieldset>

</form>

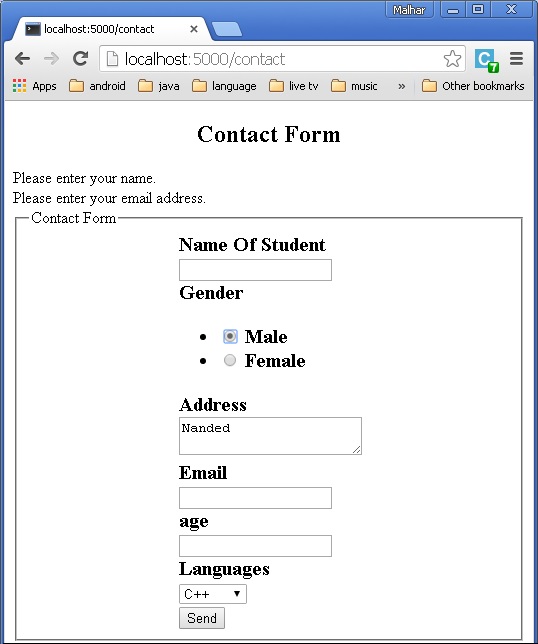
</body>

</html>

Run **formexample.py** in Python shell and visit URL **http://localhost:5000/contact**. The **Contact** form will be displayed as shown below.



If there are any errors, the page will look like this −



If there are no errors, **‘success.html’** will be rendered.



## Flask – SQLite

The HTML script for **‘student.html’** is as follows −

<html>

<body>

<form action = "{{ url\_for('addrec') }}" method = "POST">

<h3>Student Information</h3>

Name<br>

<input type = "text" name = "nm" /></br>

Address<br>

<textarea name = "add" ></textarea><br>

City<br>

<input type = "text" name = "city" /><br>

PINCODE<br>

<input type = "text" name = "pin" /><br>

<input type = "submit" value = "submit" /><br>

</form>

</body>

</html>

Finally, the **‘/’** URL rule renders a **‘home.html’** which acts as the entry point of the application.

@app.route('/')

def home():

return render\_template('home.html')

from flask import Flask, render\_template, request

import sqlite3 as sql

app = Flask(\_\_name\_\_)

@app.route('/')

def home():

return render\_template('home.html')

@app.route('/enternew')

def new\_student():

return render\_template('student.html')

@app.route('/addrec',methods = ['POST', 'GET'])

def addrec():

if request.method == 'POST':

try:

nm = request.form['nm']

addr = request.form['add']

city = request.form['city']

pin = request.form['pin']

with sql.connect("database.db") as con:

cur = con.cursor()

cur.execute("INSERT INTO students (name,addr,city,pin)

VALUES (?,?,?,?)",(nm,addr,city,pin) )

con.commit()

msg = "Record successfully added"

except:

con.rollback()

msg = "error in insert operation"

finally:

return render\_template("result.html",msg = msg)

con.close()

@app.route('/list')

def list():

con = sql.connect("database.db")

con.row\_factory = sql.Row

cur = con.cursor()

cur.execute("select \* from students")

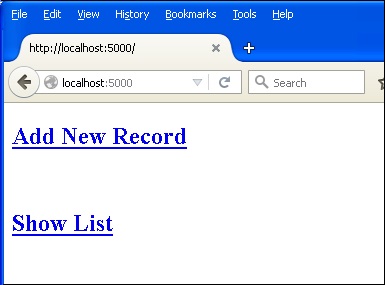
rows = cur.fetchall();

return render\_template("list.html",rows = rows)

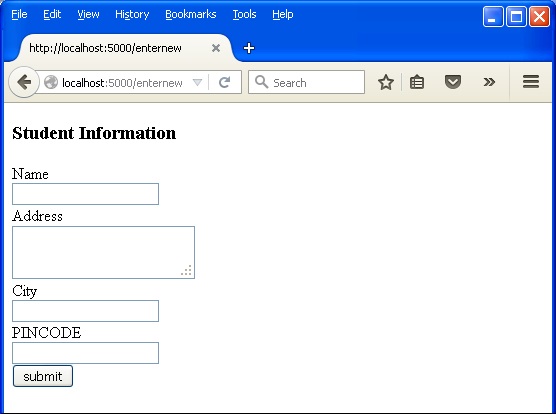
if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug = True)

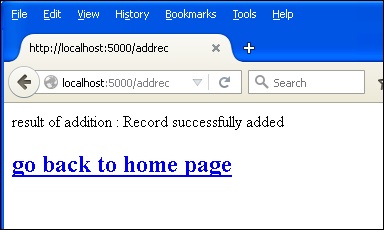
Run this script from Python shell and as the development server starts running. Visit **http://localhost:5000/** in browser which displays a simple menu like this –



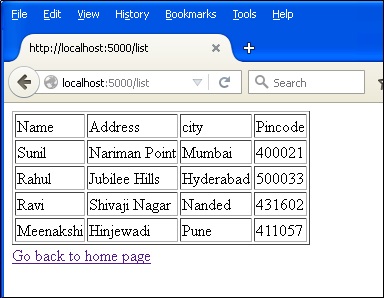
Click **‘Add New Record’** link to open the **Student Information** Form.



Fill the form fields and submit it. The underlying function inserts the record in the students table.



Go back to the home page and click **‘Show List’** link. The table showing the sample data will be displayed.



## Flask – SQLAlchemy

from flask import Flask, request, flash, url\_for, redirect, render\_template

from flask\_sqlalchemy import SQLAlchemy

app = Flask(\_\_name\_\_)

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///students.sqlite3'

app.config['SECRET\_KEY'] = "random string"

db = SQLAlchemy(app)

class students(db.Model):

id = db.Column('student\_id', db.Integer, primary\_key = True)

name = db.Column(db.String(100))

city = db.Column(db.String(50))

addr = db.Column(db.String(200))

pin = db.Column(db.String(10))

def \_\_init\_\_(self, name, city, addr,pin):

self.name = name

self.city = city

self.addr = addr

self.pin = pin

@app.route('/')

def show\_all():

return render\_template('show\_all.html', students = students.query.all() )

@app.route('/new', methods = ['GET', 'POST'])

def new():

if request.method == 'POST':

if not request.form['name'] or not request.form['city'] or not request.form['addr']:

flash('Please enter all the fields', 'error')

else:

student = students(request.form['name'], request.form['city'],

request.form['addr'], request.form['pin'])

db.session.add(student)

db.session.commit()

flash('Record was successfully added')

return redirect(url\_for('show\_all'))

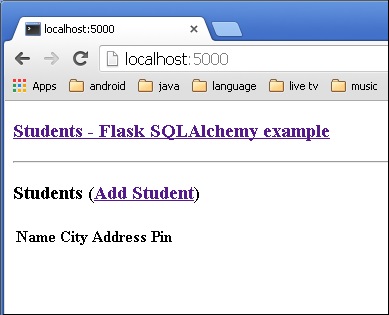
return render\_template('new.html')

if \_\_name\_\_ == '\_\_main\_\_':

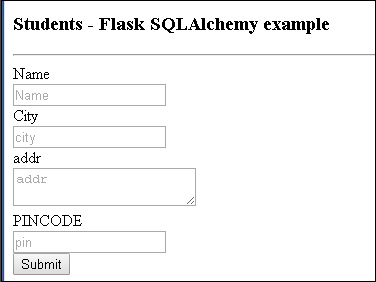
db.create\_all()

app.run(debug = True)

Run the script from Python shell and enter **http://localhost:5000/** in the browser.

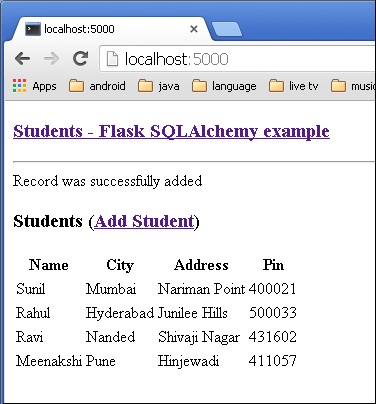


Click the **‘Add Student’** link to open **Student information** form.



Fill the form and submit. The home page reappears with the submitted data.

We can see the output as shown below.



## Flask – Sijax

import os

from flask import Flask, g

from flask\_sijax import sijax

path = os.path.join('.', os.path.dirname(\_\_file\_\_), 'static/js/sijax/')

app = Flask(\_\_name\_\_)

app.config['SIJAX\_STATIC\_PATH'] = path

app.config['SIJAX\_JSON\_URI'] = '/static/js/sijax/json2.js'

flask\_sijax.Sijax(app)

@app.route('/')

def index():

return 'Index'

@flask\_sijax.route(app, '/hello')

def hello():

def say\_hi(obj\_response):

obj\_response.alert('Hi there!')

if g.sijax.is\_sijax\_request:

# Sijax request detected - let Sijax handle it

g.sijax.register\_callback('say\_hi', say\_hi)

return g.sijax.process\_request()

return \_render\_template('sijaxexample.html')

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug = True)

When a Sijax requests (a special **jQuery.ajax()** request) to the server, this request is detected on the server by **g.sijax.is\_sijax\_request()**, in which case you let **Sijax** handle the request.

All the functions registered using **g.sijax.register\_callback()** are exposed for calling from the browser.

Calling **g.sijax.process\_request()** tells Sijax to execute the appropriate (previously registered) function and return the response to the browser.